TP2 : Entrées / Sorties

RS : Réseaux et Systèmes

Quelques conseils pour programmer en C (Rappels de CSH)

Dans ce TP, vous allez être amené à écrire des programmes C de taille moyenne. Il est conseillé d'appliquer les règles suivantes

- *Compilez avec* `-Wall -Werror`. Le compilateur détectera les erreurs d'inattention.
- *Utilisez valgrind au moindre problème.* Si vous ajouter `-g` aux options de compilation, les messages de valgrind sont encore plus utiles.
- *Indentez votre code.* Quand le programme fait plus de 50 lignes, c'est indispensable.
- *Fermez tous les descripteurs inutiles.* Sans cela, il devient très difficile de détecter les fins de flux.

★ **Exercice 1: Créer et utiliser un tube.**

Les tubes permettent habituellement à deux processus d'échanger des données. Dans un premier temps, et à but pédagogique, nous allons écrire un programme se parlant à lui-même.

▷ **Question 1:** Créez un tube, puis écrivez la chaine `"Bonjour, moi"` dans l'entrée du tube. Lisez ensuite le contenu du tube à sa sortie, et constatez que la chaîne est inchangée.

★ **Exercice 2: Déterminer la taille d'un tube.**

Pour permettre à l'écrivain de continuer avant même que le lecteur n'ai lu ces données, le système d'exploitation attache une zone de stockage à chaque tube. Les données écrites y sont stockées en attendant que le lecteur ne les réclame. L'objectif est de déterminer expérimentalement la taille de cette zone sur neptune.

▷ **Question 1:** Écrire un programme ouvrant un tube et écrivant dedans le nombre d'octets en paramètre.

▷ **Question 2:** Lancez votre programme avec un argument de plus en plus grand pour trouver la taille des tampons des tubes sur votre machine.

★ **Exercice 3: Emettre des mails depuis vos programmes.**

On veut écrire en C l'équivalent de la commande shell suivante : `echo coucou | mail <votre login>@localhost`

▷ **Question 1:** Écrire un programme vous envoyant un mail contenant la chaîne "coucou". Vous exécuterez pour cela le programme `mail`, après avoir redirigé son entrée standard sur un tube. Référez-vous au transparent du cours numéro 93 (qui détaille l'exemple du shell).

▷ **Question 2:** Faire en sorte que le programme de la question précédente vous envoie un message toutes les 3 secondes (sans utiliser `sleep(3)`, mais en utilisant une alarme).

★ **Exercice 4: Réimplémenter popen.**

La fonction `popen` permet de lire la sortie standard d'une commande (ou d'écrire sur son entrée standard) comme s'il s'agissait d'un fichier. Exemple d'ouverture en lecture : `fich = popen("ls -lR","r");`

▷ **Question 1:** Implémentez une fonction permettant de lire la sortie d'une commande (ie, en supposant que le second argument est "r").
REMARQUE : popen() retourne un `FILE*` utilisable avec `fscanf`. Votre version devra retourner un descripteur pour `read` (c'est plus simple ainsi).

★ **Exercice 5: Communication inter-processus.** (mini-projet)

Observez le programme `/home/depot/2A/RS/distributeur.c`. Il lit le flux de caractères arrivant sur l'entrée standard en séparant les chiffres des lettres, effectue l'opération appropriée en fonction du type de caractère (sommer les chiffres ; réaliser un spectre de fréquence pour les lettres) et enfin affiche le résultat.

Pour cela, le programme est compsé de trois entités :

- un distributeur (en charge de la répartition des caractères) ;
- un additionneur (opérant sur les chiffres) ;
- un compteur (opérant sur les lettres).

▷ **Question 1:** Adapter ce programme pour que les fonctions d'additionneur et de compteurs soient assurées par des processus enfants d'un processus père qui assure la fonction de distributeur. Les communications entre les processus se font par tube.